WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
searc$4 near5 clas$2 near4 cache	9

Database:
- US Patents Full-Text Database
- US Pre-Grant Publication Full-Text Database
- JPO Abstracts Database
- EPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Search: L7

Refine Search

Recall Text | Clear

Search History

DATE: Friday, February 15, 2002 Printable Copy Create Case

DB=USPT; PLUR=YES; OP=OR

Set Name side by side	Query	Hit Count	Set Name result set
L7	searc$4 near5 clas$2 near4 cache	9	L7
L6	L5 and searc$3	12	L6
L5	L4 and clas$4 near8 path	27	L5
L4	(generat$4 or creat$4) near5 cache	3541	L4
L3	(generat$4 or creat$4) near5 cache near9 clas$4 near5 path and search	1	L3
L2	L1 and Java	1	L2
L1	cache near4 clas$3 near9 search	9	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 10 of 12 returned.

☐ 1. Document ID: US 6272650 B1

L6: Entry 1 of 12 File: USPT Aug 7, 2001

US-PAT-NO: 6272650
DOCUMENT-IDENTIFIER: US 6272650 B1

TITLE: System and method for disambiguating scene graph loads

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 2. Document ID: US 6263496 B1

L6: Entry 2 of 12 File: USPT Jul 17, 2001

US-PAT-NO: 6263496
DOCUMENT-IDENTIFIER: US 6263496 B1

TITLE: Self modifying scene graph

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 3. Document ID: US 6243856 B1

L6: Entry 3 of 12 File: USPT Jun 5, 2001

US-PAT-NO: 6243856
DOCUMENT-IDENTIFIER: US 6243856 B1

TITLE: System and method for encoding a scene graph

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 4. Document ID: US 6192043 B1

L6: Entry 4 of 12 File: USPT Feb 20, 2001

US-PAT-NO: 6192043

DOCUMENT-IDENTIFIER: US 6192043 B1

TITLE: Method of caching routes in asynchronous transfer mode PNNI networks

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 5. Document ID: US 6115547 A

L6: Entry 5 of 12 File: USPT Sep 5, 2000

US-PAT-NO: 6115547
DOCUMENT-IDENTIFIER: US 6115547 A

TITLE: Flash configuration cache

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 6. Document ID: US 5940877 A

L6: Entry 6 of 12 File: USPT Aug 17, 1999

US-PAT-NO: 5940877
DOCUMENT-IDENTIFIER: US 5940877 A

TITLE: Cache address generation with and without carry-in

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 7. Document ID: US 5537574 A

L6: Entry 7 of 12 File: USPT Jul 16, 1996

US-PAT-NO: 5537574
DOCUMENT-IDENTIFIER: US 5537574 A

TITLE: Sysplex shared data coherency method

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 8. Document ID: US 5418922 A

L6: Entry 8 of 12 File: USPT May 23, 1995

US-PAT-NO: 5418922
DOCUMENT-IDENTIFIER: US 5418922 A

TITLE: History table for set prediction for accessing a set associative cache

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 9. Document ID: US 5392410 A
L6: Entry 9 of 12 File: USPT Feb 21, 1995

US-PAT-NO: 5392410
DOCUMENT-IDENTIFIER: US 5392410 A

TITLE: History table for prediction of virtual address translation for cache access

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 10. Document ID: US 5335325 A
L6: Entry 10 of 12 File: USPT Aug 2, 1994

US-PAT-NO: 5335325
DOCUMENT-IDENTIFIER: US 5335325 A

TITLE: High-speed packet switching apparatus and method

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

Generate Collection | Print

Terms	Documents
L5 and searc$3	12

Display Format: TI | Change Format

Previous Page Next Page

WEST

Help Logout Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
L5 and searc$3	12

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L6

Refine Search

Recall Text Clear

Search History

DATE: Friday, February 15, 2002 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=OR		
L6	L5 and searc$3	12	L6
L5	L4 and clas$4 near8 path	27	L5
L4	(generat$4 or creat$4) near5 cache	3541	L4
L3	(generat$4 or creat$4) near5 cache near9 clas$4 near5 path and search	1	L3
L2	L1 and Java	1	L2
L1	cache near4 clas$3 near9 search	9	L1

END OF SEARCH HISTORY

WEST

End of Result Set

Generate Collection Print

L2: Entry 1 of 1 File: USPT Jul 6, 1999

DOCUMENT-IDENTIFIER: US 5920725 A
TITLE: Run-time object-synthesis and transparent client/server updating of distributed objects using a meta server of all object descriptors

Brief Summary Paragraph Right (3):
Distributed computing has reduced the computing burden on central servers by partitioning software applications across server and client machines. At first, client PC's merely executed viewer or "browser" applications to view documents and data from the servers. Small program routines such as common-gateway-interface (cgi) scripts were executed on the server for the client's browser. Later, browser add-on programs or applets written in Java or ActiveX were downloaded from the server to the browser-client and executed on the user's PC. These relatively small client applets are appropriately known as thin clients.

Brief Summary Paragraph Right (20):
Multi-file applications written in modern languages such as C++ and Java are typically compiled on a client and uploaded and re-installed to a server machine, or compiled and loaded on a server. Distributed programming such as DCOM use temporary program classes known as object factories to generate object instances, but these object factories still require programming effort and are not interactive.

Detailed Description Paragraph Right (46):
Invalid bits are part of each object instance. The client cache contains an index of all object instances of a particular class. This provides a quick mechanism for the object adaptor to invalidate objects, as the object adaptor need only command the client cache to invalidate a class. The client cache uses its index to locate each object instance for the invalidated class. The client cache then sets invalid bits for each objects instance for the classes being updated. The object instance itself in the client's memory does not have to be directly invalidated by the object adaptor. This saves computational work since objects can reside at different addresses, saving the object adaptor from performing address lookups or translation. The object adaptor merely has to send notification with a list of invalid object classes to the client caches. Each client cache then searches for objects and classes from the notification list, and invalidates these objects.

Detailed Description Paragraph Right (55):
The first statement in the list of API calls 128 instructs the meta

server to make a new attribute. The attribute's name, type, and string length are specified in the next three lines. The final line of Java code:

WEST

Generate Collection Print

L1: Entry 3 of 9 File: USPT Sep 28, 1999

DOCUMENT-IDENTIFIER: US 5960197 A
TITLE: Compiler dispatch function for object-oriented C

Brief Summary Paragraph Type 1 (6):
Step 3c: Search the class cache identified in step 3b for the method name string whose address is loaded in the second specified register (see step 2 above). If a match between the method name string and an entry in the class cache is not found, the current class's class data structure is searched and then the current class's parent class data structure is searched. Each class data structure in a method call's inheritance chain is searched as needed in this manner. If no match is found, a run-time error is generated. If a match is found, the current class cache data structure is updated and processing continues to step 3d.

Detailed Description Paragraph Right (14):
In those instances where a method's executable code is defined at compile time, it is possible for a compiler to use a direct method invocation process. When this is possible, a method dispatch technique in accordance with the invention eliminates the need for costly (time consuming) address comparison operations and searches through one or more class (or similar) cache data structures. Since this information is known at compile time for a majority of method calls in a typical object-oriented program (e.g., source-code), use of a direct dispatch technique can provide a substantial speed gain during program execution. On the other hand, if a method's executable code segment is undefined at compilation time, no degradation in performance over existing (dynamic) method dispatch techniques is incurred.

WEST

Generate Collection Print

L6: Entry 4 of 12 File: USPT Feb 20, 2001

DOCUMENT-IDENTIFIER: US 6192043 B1
TITLE: Method of caching routes in asynchronous transfer mode PNNI networks

Brief Summary Paragraph Right (25):
A flow diagram illustrating the prior art method of network routing using the well known Dijkstra method is shown in FIG. 2. The first step is to perform an address lookup (step 20) in the node's topology database on the advertised reachable prefixes and addresses to find all the nodes that advertise a `BEST MATCH` to the destination node (step 22). Each of the nodes found in the search are marked as a `destination` for this particular route calculation (step 24). Note that more than one node can be marked as a destination node.

Brief Summary Paragraph Right (39):
There is therefore provided in accordance with the present invention, in a network based on the Private Network to Network Interface (PNNI) standard and consisting of a plurality of nodes, a method of caching routes generated using the Dijkstra algorithm into a cache maintained on each node, the Dijkstra algorithm utilizing a PATH list in the calculation of routes, the method supporting a single class of call, the method comprising the steps of maintaining a cache sequence number on each node, maintaining a global cache sequence count on each node, setting the cache sequence number equal to the cache sequence count when the node is placed onto the PATH list and constructing a routing list directly from parent pointers, defining a route from a destination node to a local node, if the cache sequence number equals the cache sequence count.

Brief Summary Paragraph Right (41):
There is also provided in accordance with the present invention, in a network based on the Private Network to Network Interface (PNNI) standard and consisting of a plurality of nodes, a method of caching routes generated using the Dijkstra algorithm into a plurality of caches maintained on each node, each cache supporting a different class of call, the Dijkstra algorithm utilizing a PATH list in the calculation of routes, the method comprising the steps of maintaining N sets of variables consisting of a cache sequence number, a parent pointer and a global cache sequence count for each node wherein each set of variables is associated with a single class of call to be supported, setting the cache sequence number, within a particular set of variables corresponding to a particular class of call, equal to the cache sequence count when the node is placed onto the PATH list and constructing a routing list directly from parent pointers, defining a route from a destination node to a local node, if the cache sequence

number, within a particular set of variables corresponding to a particular class of call, equals the cache sequence count within the set of variables thereof.

Detailed Description Paragraph Right (5):
A flow diagram illustrating the initialization portion of the caching method of the present invention is shown in FIG. 6. The global variable CACHE_SEQ_CNT is created and initialized to one (step 50). Next, the field CACHE_SEQ_NUM in each node is initialized to zero (step 52). The field CACHE_SEQ_NUM is initialized to zero whenever a node descriptor is initialized.

Detailed Description Paragraph Right (9):
If the global variable CACHE_SEQ_CNT and the node descriptor field CACHE_SEQ_NUM are not equal then the method proceeds as follows. Each of the nodes found in the search are marked as a `destination` for this particular route calculation (step 86). Note that more than one node can be marked as a destination node.

CLAIMS:

1. In a network based on the Private Network to Network Interface (PNNI) standard and consisting of a plurality of nodes, a method of caching routes generated using the Dijkstra algorithm into a cache maintained on each node, said Dijkstra algorithm utilizing a PATH list in the calculation of routes, said method supporting a single class of call, said method comprising the steps of:

maintaining a cache sequence number on each node;

maintaining a global cache sequence count on each node;

setting said cache sequence number equal to said cache sequence count when the node is placed onto said PATH list; and

constructing a routing list directly from parent pointers, defining a route from a destination node to a local node, if said cache sequence number equals said cache sequence count.

6. In a network based on the Private Network to Network Interface (PNNI) standard and consisting of a plurality of nodes, a method of caching routes generated using the Dijkstra algorithm into a plurality of caches maintained on each node, each cache supporting a different class of call, said Dijkstra algorithm utilizing a PATH list in the calculation of routes, said method comprising the steps of:

maintaining N sets of variables consisting of a cache sequence number, a parent pointer and a global cache sequence count for each node wherein each set of variables is associated with a single class of call to be supported;

setting said cache sequence number, within a particular set of variables corresponding to a particular class of call, equal to said cache sequence count when the node is placed onto said PATH list; and

constructing a routing list directly from parent pointers, defining a route from a destination node to a local node, if said cache sequence number, within a particular set of variables corresponding to a particular class of call, equals said cache sequence count within the set of variables thereof.

WEST

☐ Generate Collection Print

L7: Entry 1 of 9 File: USPT Jul 10, 2001

DOCUMENT-IDENTIFIER: US 6260045 B1
TITLE: Method and apparatus for optimizing interface dispatching in an object-oriented programming environment

Detailed Description Paragraph Right (12):
According to one embodiment, if the class implements only one interface, then that entry is placed directly into the cache, and the class list that is searched when the cache is missed is set to zero. Thus, the class interface cache is pre-primed, which avoids the initial priming during run time execution. Additionally, according to an alternate embodiment, by removing the implemented interface from the list, the negative search case (when the programmer tests if an object is not an instance of an interface) is helped as well. Many interfaces are implemented by custom classes that have been created just to support that interface. This increases the likelihood that a class will implement only one interface.